Project Feedback - Brett Bailey

(a) What did you learn from this project?

This project afforded the opportunity to familiarize myself with the Altera FPGA tools, having previously only worked with Xilinx parts and tools. I also got to play with synthesis attributes to specify how synthesis processes VHDL and infers components, as well as learned how ModelSim completely ignores these synthesis attributes, which can cause divergence between the simulated and synthesized design. I explored different VHDL RTL description paradigms while trying to make the compilation tools infer as many of the control signals as humanly possible.

(b) What would you do differently next time?

I'd try to find a task that regular processors typically don't do very well but that FPGAs can do fairly easily, then make an instruction in my processor that did that task. This would create a purpose built processing unit with the added benefit of being able to do more general purpose processing as well. Maybe some kind of SIMD task with a large quantity of registers. I'd also create some kind of useful program to demonstrate the capabilities and unique features of the processor instead of my current test program that ensures that all the instructions work properly without any of the instructions having a larger purpose.

(c) What is your advice to someone who is going to work on a similar project?

Produce an automated test bench, at least at the system level. It doesn't have to figure out why something broke, only tell you when so you can look at that specific area on the waveform to determine the error. Once there are many signals to deal with and many possible instructions, it is easy to introduce an error and not notice without automated error checking. It also allows you to implement updates and quickly check that they didn't break anything.

Don't worry overmuch about creating the fastest running processor. Optimizing for speed has been done better than you'll be able to do and a thousand times before. Too much speed also introduces various limitations in how you build your processor and the range of operations that instructions can do. Be creative by designing a processor that does something that other processors don't, even if it runs slowly.

I also advocate doing the project on your own, giving you complete creative control and forcing you to master all the design and implementation aspects instead of only select portions. You'll need to know the whole process if you ever plan to do hardware design for a living.

Lastly, I'd suggest that you ignore the "do a little of the design every week" thing. Creating a new report every week is one thing, but fragmenting the design over such a long time can really damage the quality, especially if you have to jump through hoops to test your system because you don't have a memory implemented. Waiting so long to completely finish implementing the last of the system also reduces the time available for verification and tightening up timing.